Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

В.Г. Резник

# РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Электронный курс

Томск
2020

УДК 004.75(075.8)

ББК 32.973.202я73

Р 344

**Резник, Виталий Григорьевич**

Р 344 Распределённые вычислительные системы: электронный курс / В.Г. Резник. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2020.

Дисциплина «Распределённые вычислительные системы» (РВС) входит в группу модулей «Лабораторно-экзаменационная сессия (09.03.01, профиль ПОСВТиАС)» дисциплин ФДО ТУСУР, обеспечивающих промежуточную аттестацию студентов в виде лабораторной экзаменационной сессии (ЛЭС).

Электронный курс предназначен для информационной и организационной поддержки изучения дисциплины РВС студентов бакалавриата ФДО ТУСУР, проходящих обучение по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

УДК 004.75

# Оглавление

# Введение

Электронный курс предназначен для информационной и организационной поддержки изучения дисциплины «Распределённые вычислительные системы» (РВС) студентов бакалавриата ФДО ТУСУР, проходящих обучение по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

Дисциплина РВС входит в группу модулей «Лабораторно-экзаменационная сессия (09.03.01, профиль ПОСВТиАС)» дисциплин ФДО ТУСУР, обеспечивающих промежуточную аттестацию студентов в виде лабораторной экзаменационной сессии (ЛЭС).

Учебный материал дисциплины РВС разработан на кафедре АСУ и предназначен для обучения студентов бакалавриата по направлению подготовки 09.03.01 «Информатика и вычислительная техника», в общем объёме 216 часов учебных занятий. Методика процесса обучения предполагает использование материально-технической базы кафедры АСУ ТУСУР и специального программного обеспечения, размещённого на компьютерах учебных классов кафедры АСУ для выполнения студентами необходимых лабораторных работ.

В результате процесса обучения заявленной дисциплины, студенты должны получить следующие компетенции:

1. ОПК-3 — Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учётом основных требований информационной безопасности.
2. ОПК-9 — Способен осваивать методики использования программных средств для решения практических задач.
3. ПКС-1 — Способен заниматься профессиональной разработкой программного обеспечения и принимать проектные решения при выполнении производственных и научно-исследовательских задач.

По результатам процесса обучения студенты проходят промежуточную аттестацию в виде дифференцированного зачёта (зачёта с оценкой).

Для организации процесса обучения студентов ФДО ТУСУР на сайте кафедры АСУ выделен специальный каталог «Распределённые вычислительные системы», предназначенный для размещения всего необходимого студентам учебного материала и доступного по указанной ссылке из личных кабинетов студентов.

Непосредственно по видам учебной деятельности студенты ФДО выполняют:

1. Лабораторные занятия — **144 часа**.
2. Самостоятельная работа — **62 часа**.
3. Самостоятельная работа под руководством преподавателя — **6 часов**.
4. Подготовка и сдача зачёта с оценкой — **4 часа**.

Для реализации указанного выше плана учебных работ заявленный электронный курс предоставляет информацию изложенную в следующих разделах пособия:

1. Общие вопросы организации учебного процесса.
2. Самостоятельное изучение теоретического материала и выполнение лабораторных работ.
3. Проведение лабораторной экзаменационной сессии (ЛЭС).

# 1 Общие вопросы организации учебного процесса

Дисциплина «Распределённые вычислительные системы» (РВС), изучаемая студентами «Факультета дистанционного обучения» (ФДО) ТУСУР, относится к группе учебных предметов, для которых предусмотрена лабораторная экзаменационная сессия (ЛЭС).

Обоснование выбранной технологии обучения обусловлено необходимостью использования, в самом процессе изучения дисциплины, достаточно большого количества различного системного и инструментального программного обеспечения ЭВМ, которое затруднительно самостоятельно установить студенту на собственные компьютеры. По этой причине, основной инфраструктурой для проведения лабораторных работ выбраны учебные классы кафедры АСУ ТУСУР, обеспеченные необходимыми аппаратными и программными средствами изучения данной дисциплины, в соответствии с заявленными компетенциями.

Поскольку технология дистанционного обучения требует от студента значительной части самостоятельной работы без участия преподавателя, поэтому данный раздел уделяет основное внимание двум организационным аспектам самого процесса обучения:

а) перечню и назначению учебного материала изучаемой дисциплины;

б) организации лабораторной экзаменационной сессии (ЛЭС).

Такой подход должен настроить студента на правильный самоконтроль своих учебных достижений.

## 1.1 Перечень и назначение учебного материала дисциплины

Для целей организации учебного процесса на сайте кафедры АСУ выделен специальный каталог «Распределённые вычислительные системы», предназначенный для размещения всего необходимого студентам учебного материала. Этот каталог доступен студентам из их личных кабинетов, по прямой ссылке. В этом каталоге так же находится файл данного электронного курса.

Студенту следует внимательно изучить назначение каждого учебного пособия, чтобы в дальнейшем эффективно использовать его как в самостоятельной работе, так и во время прохождения ЛЭС. Более подробное применение указанных пособий изложено во втором разделе данного курса. Здесь же приведён только их полный список с краткой характеристикой прикладного назначения:

1. Учебное пособие «Распределённые вычислительные сети» является основной литературой, содержащей весь теоретический материал необходимый студенту для изучения дисциплины РВС и сдачи дифференцированного зачёта. Текст учебного материала содержит примеры его практического применения, что необходимо для успешного выполнения всех лабораторных работ.

2. Учебно-методическое пособие «Самостоятельная и индивидуальная работа студента по направлению подготовки бакалавра 09.03.01» является обязательной литературой, в которой перечислены все разделы изучаемой дисциплины, их тематическое назначение, перечень вопросов для аттестации студентов и названия лабораторных работ, привязанных к учебному материалу соответствующих разделов основной литературы. Основное назначение данного пособия — облегчить студенту подготовку к промежуточной аттестации, поскольку каждый аттестационный вопрос именован так, что имеет соответствующее название в учебном пособии основной литературы.

3. Учебно-методическое пособие «Лабораторные работы по направлению подготовки бакалавриата 09.03.01» является обязательной литературой, в которой кратко описаны требования для выполнения лабораторных работ. Текст учебного материала этого по-

собия опирается на знание учебного материала основной литературы.

4. Учебно-методическое пособие «Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux» — общее для многих дисциплин пособие, содержащее учебный материал по содержанию и практическому применению системного и инструментального программного обеспечения ЭВМ, установленного в учебных классах кафедры АСУ. Изучение и успешное практическое применение учебного материала этого пособия является основой для успешного выполнения лабораторных работ изучаемой дисциплины.

## 1.2 Организация лабораторной экзаменационной сессии (ЛЭС)

Все основные организационные аспекты проведения лабораторной экзаменационной сессии (ЛЭС) осуществляются руководством ФДО ТУСУР. К ним относятся:

1. Определение сроков проведения сессии, количество и состав учебных групп студентов, допущенных к обучению.
2. Составление общего расписания занятий для всех дисциплин, входящих перечень модулей ЛЭС и согласование этого расписания с преподавателями кафедры АСУ, ответственными за проведение конкретных занятий.
3. Проведение начальных и текущих организационных собраний со студентами.
4. Подведение итогов проведения ЛЭС.

Преподаватель, назначенный для проведения занятий по дисциплине РВС, организует проведение учебного процесса ЛЭС согласно предоставленного ему руководством ФДО общего расписания занятий.

В список организационных вопросов преподавателя дисциплины РВС входит:

1. Размещение учебных и информационных материалов дисциплины согласно профилю модулей «Лабораторно-экзаменационная сессия (09.03.01, профиль ПОСВТиАС)».
2. Проведение лекционных и лабораторных занятий с группами студентов согласно назначенному ему расписанию.
3. Оценку текущих и итоговых результатов работы студентов.
4. Передачу итоговых результатов оценивания студентов руководству ФДО ТУСУР.

# 2 Самостоятельное изучение теоретического материала и выполнение лабораторных работ

Дистанционное обучение студента предполагает значительный объем времени затрачивать на самостоятельное изучение дисциплины РВС. Настоящий раздел излагает методические рекомендации по организации самостоятельного процесса усвоения предоставленного ему учебного материала дисциплины.

Поскольку учебный план дисциплины предполагает значительный объем лабораторных работ, требующих сложного системного и инструментального программного обеспечения ЭВМ, то, в плане методических рекомендаций, следует разделить процессы изучения теоретического материала дисциплины и выполнение требуемых лабораторных работ.

## 2.1 Изучение теоретического материала дисциплины

Полный объём необходимого теоретического материала дисциплины РВС изложен в учебном пособии «Распределённые вычислительные сети». Это пособие является основной литературой, содержащей также программные примеры необходимые для выполнения всех лабораторных работ.

Общий и текущий контроль качества изученного материала рекомендуется выполнять посредством вопросов, изложенных в учебно-методическом пособии «Самостоятельная и индивидуальная работа студента по направлению подготовки бакалавра 09.03.01». Такой подход обеспечит необходимый уровень общей теоретической подготовки студента.

Весь учебный материал дисциплины последовательно изложен в пяти разделах, которые необходимо изучать строго последовательно. Такой подход максимально оптимизирует понимание и последующее применение полученных знаний.

*Первый раздел* «Введение в теорию вычислительных сетей» является введением в предметную область изучаемой дисциплины. Здесь обсуждаются базовые концепции, положенные в основы современной теории распределённых систем. Для этого приводится анализ классификации систем обработки данных (СОД), вводятся необходимые определения и даётся начальный обзор технологий, рассматриваемых более подробно в следующих разделах учебного пособия.

> **Замечание** — Знание студентом всех вопросов, связанных с первым разделом, является обязательным требованием для его положительной аттестации.

*Второй раздел* «Инструментальные средства языка Java» полностью посвящён базовым основам синтаксиса и семантики языка Java. Учебная цель данного раздела — обеспечение студентов базовыми знаниями объектно-ориентированного языка, широко используемого в современной практике создания распределённых систем. К базовым знаниям здесь относятся: общие вопросы программирования на языке Java, включая средства управления сетевыми соединениями и программный доступ к базам данных.

Уровень изложения указанного материала выбран с учётом знания студентом основ программирования на языках C/C++, а также опыта работы с сетями и базами данных, полученного им при изучении соответствующих дисциплин. Практические примеры данного раздела ориентированы на последующее применение полученных знаний и навыков при освоении учебного материала последующих разделов.

**Замечание** — Первые два раздела завершают подготовительную часть учебного материала изучаемой дисциплины.

*Третий раздел* «Объектные распределенные системы» раскрывает тему объектных распределённых систем, заложивших теоретическую и практическую основу изучаемой дисциплины. Результаты теории объектного подхода, представленного описанием технологий CORBA и RMI, определили класс *сильносвязанных* распределённых систем, отделив от теории РВС большой набор предшествующих классических технологий, таких как распределённые компьютерные среды (DCE) и удалённый вызов процедур (RPC).

*Четвёртый раздел* «Web-технологии распределённых систем» посвящён современной тематике технологий Интернет, основанной на протоколе HTTP. С момента своего появления эти технологии стали претендовать на самостоятельное направление реализации распределённых систем. Во многом это связано и с практическими технологическими достижениями языка Java, например, реализацию технологий сервлетов и JSP-страниц, а также успешную разработку инструментального программного обеспечения, например, в виде системы Java Enterprise Edition.

**Замечание** — Изложение учебного материала представлено как самостоятельное технологическое направление, которое не тождественно более узкому технологическому направлению РВС. Тем не менее первые практические реализации *слабосвязанных распределённых систем* были основаны на технологиях Web.

*Пятый раздел* «Сервис-ориентированные архитектуры» кратко описывает концепцию *слабосвязанных* архитектур, таких как: концепция SOA, облачные технологии и вычисления GRID.

**Замечание** — В настоящее время теория и практика создания сервис-ориентированных систем сформировались в отдельное научное направление со своей парадигмой, классификацией и широким набором методов их реализации. По этой причине развёрнутое изучение этой тематики излагается в соответствующем магистерском курсе по направлению 09.04.01 «Информатика и вычислительная техника».

## 2.2 Выполнение лабораторных работ

Формально обязательным учебно-методическим пособием, предназначенным для выполнения всех девяти лабораторных работ, является учебный материал «Лабораторные работы по направлению подготовки бакалавриата 09.03.01». Тем не менее структура и содержание лабораторных работ тематически привязана к основной литературе, представленной учебным пособием «Распределённые вычислительные сети». Именно в таком тандеме и следует самостоятельно выполнять все лабораторные работы. Дополнительно, перечень лабораторных работ, с привязкой разделам основной литературы, изложен в пособии «Самостоятельная и индивидуальная работа студента по направлению подготовки бакалавра 09.03.01».

Чтобы отразить индивидуальные особенности выполнения каждой лабораторной работы, приведём их краткие характеристики.

*Лабораторная работа №1* «Тестирование ПО рабочей области студента» соответствует *первому разделу основной литературы*, но фактически не использует её учебный материал, поскольку является стандартным учебным занятием связанным с изучением системного и инструментального программного обеспечения, предназначенного для организации выполнения последующих работ.

Формально, используя инфраструктуру учебных классов кафедры АСУ, студент должен проверить переданные ему индивидуальные программные средства и показать успешные навыки работы с ними. Но проблема возникает, когда студент желает выполнить лабораторную работу на собственном компьютере, да ещё — в среде ОС MS Windows.

Чтобы минимизировать проблемы выполнения лабораторных работ, на кафедре АСУ собран и используется специальный дистрибутив ОС Linux (ОС УПК АСУ), который может устанавливаться в файловую систему ОС MS Windows. Для описания того, как это делается предназначено учебно-методическое пособие «Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux». В этом же пособии указан сайт кафедры АСУ, с которого студент может самостоятельно скачать дистрибутив рекомендуемой ОС.

*Лабораторные работы №2 - №5* предназначены для практического закрепления учебного материала, изложенного во *втором разделе основной литературы*. Учебная цель этих работ — освоить теорию и практику применения языка для решения базовых программистских задач, включающих задачи разработки простейших сетевых приложений и работу с базами данных. Сами названия этих работ достаточно полно раскрывают их прикладное назначение:

а) Лабораторная работа №2 — «Дистрибутив языка Java и среда разработки Eclipse EE».

б) Лабораторная работа №3 — «Базовые средства и ввод-вывод языка Java».

в) Лабораторная работа №4 — «Сокеты и сетевое ПО языка Java».

г) Лабораторная работа №5 — «Технология работы с базами данных».

*Лабораторная работа №6* «Реализация распределенной системы средствами технологии RMI» соответствует *третьему разделу основной литературы*. Учебная цель данной работы — закрепление на практике классического варианта технологии создания сильносвязанных распределённых систем инструментальными средствами языка Java.

**Замечание** — Следует обратить внимание, что применение технологии RMI положено в тематическую основу проведения ЛЭС по изучаемой дисциплине.

*Лабораторные работы №7 - №9* предназначены для практического закрепления учебного материала, изложенного во *четвёртом разделе основной литературы*. Учебная цель этих работ — освоить теорию и практику применения современных web-технологий.

Последние три работы завершают перечень лабораторных занятий по изучаемой дисциплине. Перечень их названий достаточно полно характеризует прикладные аспекты закрепляемых знаний:

а) Лабораторная работа №7 — «Технология сервлетов на базе сервера Apache Tomcat».

б) Лабораторная работа №8 — «Технология JSP для формирования динамических HTML-страниц».

в) Лабораторная работа №9 — «Шаблон проектирования MVC».

# 3 Проведение лабораторной экзаменационной сессии (ЛЭС)

Как уже было отмечено в первом разделе этого пособия, все основные организационные аспекты проведения лабораторной экзаменационной сессии осуществляются руководством ФДО ТУСУР и осуществляются в формате «Лабораторно-экзаменационная сессия (09.03.01, профиль ПОСВТиАС)».

Настоящий раздел посвящён практическим вопросам проведения ЛЭС для групп студентов, заявленных руководством ФДО ТУСУР. Содержание этой части электронного курса разделено на четыре подраздела:

1. Общие вопросы проведения работ экзаменационной сессии (ЛЭС).
2. Пример выполнения лабораторного задания №1.
3. Пример выполнения лабораторного задания №2.
4. Оценка выполнения студентом сессии ЛЭС.

## 3.1 Общие вопросы проведения работ экзаменационной сессии (ЛЭС)

Лабораторная экзаменационная сессия (ЛЭС), согласно требованиям Рабочей программы, относится к следующим видам учебной деятельности:

1) Самостоятельная работа под руководством преподавателя.
2) Подготовка и сдача зачёта с оценкой.

В соответствии с переданным на кафедру АСУ расписанием, преподаватель, руководящий обучением по данной дисциплине, заполняет электронный журнал, определённый форматом «Лабораторно-экзаменационная сессия (09.03.01, профиль ПОСВТиАС)», где:

а) размещает необходимые учебные пособия;
б) расписание проведения лекционных занятий:
в) расписание проведения контрольных занятий и занятий для проведения итоговой индивидуальной аттестации студентов.

Для проведения итоговой контрольной аттестации, студентам выдаётся общее задание на реализацию простейшей распределённой системы, в виде следующих условий:

1. Реализовать собственное произвольное распределенное приложение, содержащее программы *RMI-клиента* и *RMI-сервера*, используя теоретическую часть и примеры, изложенные в подразделе «3.3 Технология RMI» учебного пособия основной литературы «Распределённые вычислительные сети».
2. Изложить результаты индивидуальной работы в едином отчёте, шаблон и структура которого представлены в файле Отчёт_РВС_КЛП.odt.
3. Разместить итоговый отчёт в электронном журнале изучаемой дисциплины.

**Замечание** — Последующие два подраздела данного пособия содержат учебные примеры реализации и оформления результатов контрольного задания ЛЭС.

## 3.2 Пример выполнения лабораторного задания №1

**Лабораторная работа №1** содержит описание и реализацию примера ***серверной части распределенного приложения***, демонстрирующего применение технологии RMI, которая в настоящее время входит в состав пакета Java SE: JDK SE (инструментального пакета языка Java уровня Standard Edition).

Любое распределенное приложение базируется общей модели «Клиент — Сервер», что обычно предполагает размещение минимум двух приложений на разных ЭВМ, которые могут взаимодействовать между собой посредством сетевого программного обеспечения:

- ***приложение-сервер*** концентрирует в себе некоторую функциональность, которая отсутствует у ***приложения-клиента***, но необходима ему для реализации своей функциональности;
- ***приложение-клиент*** обычно реализует некоторую функциональность, которая необходима конечному пользователю, но, в силу ограниченности своих функциональных ресурсов, вынуждена обращаться за соответствующими ресурсами к одному или многим серверам.

**Замечание** — Реализуя своё собственное распределенное приложение, студент должен чётко себе представлять, какую функциональность будет выполнять приложение-клиент, а какую приложение-сервер.

В любом случае работу ***необходимо начинать с реализации приложения-сервера***, что и демонстрируется в данном учебно-методическом пособии.

Следуя общему заданию лабораторного практикума данная глава пособия описывает процесс разработки приложения-сервера, который основан на объектном подходе реализации распределенных систем. Более того, учитывая многие технологические сложности инструментальных средств технологии CORBA (см. содержание раздела 3 учебного пособия «Распределённые вычислительные сети»), студенту предоставляется возможность использования технологии RMI, которая максимально упрощена в плане своей реализации.

В методическом плане, данная глава разделена на две части:

1) описание функциональности примера приложения-сервера;
2) пример реализации RMI-сервера.

### 3.2.1 Формулировка задания на выполнение работы №1

В этом подразделе студенту следует:

1) дать формальное описание функциональных возможностей RMI-сервера;
2) описать интерфейс удалённого RMI-объекта, которым в дальнейшем могут воспользоваться приложения-клиенты;
3) описать реализацию RMI-объекта, обеспечивающего функциональность заданного интерфейса.

Далее приводится демонстрация конкретного упрощённого примера, на основе которого студент должен придумать и реализовать свой пример.

#### 3.2.1.1 Описание функциональности RMI-сервера

Общая функциональность примера RMI-сервера задаётся его свойствами:

1) сервер предоставляет приложениям-клиентам единственный метод с именем ***getMessage***, с помощью которого клиент может передать серверу произвольное текстовое сообщение;
2) сервер сохраняет принятое сообщение ***как последнее принятое***;
3) в общем случае, сервер сохраняет следующие данные: ***количество*** принятых сообщений, ***последнее*** и ***предпоследнее*** принятые сообщения;
4) с помощью метода ***getMessage***, сервер возвращает приложению-клиента массив из трёх строк: ***количество*** принятых сообщений, ***последнее*** и ***предпоследнее*** принятые сообщения.

#### 3.2.1.2 Описание интерфейса удалённого RMI-объекта на языке Java

Проводим описание интерфейса удалённого RMI-объекта, показанного на листинге 3.1, при следующих условиях:

1) интерфейс имеет имя ***Klp2Rmi***;
2) интерфейс определяется в пакете ***klp2.rmi***;
3) интерфейс расширяет интерфейс ***java.rmi.Remote***.

*Листинг 3.1 – Исходный текст интерфейса Klp2Rmi*

```
package klp2.rmi;

import java.rmi.Remote;

public interface Klp2Rmi extends Remote {
	/**
	 * Объявление метода GetMessage
	 */
	public String[] getMessage(String str) throws
		java.rmi.RemoteException;
}
```

Обратите внимание, что в дальнейшем данный интерфейс будет использоваться как приложением RMI-сервера, так и приложением RMI-клиента.

Напоминаю, что студент должен сделать своё описание интерфейса, согласно собственному описанию функциональности RMI-сервера.

#### 3.2.1.3 Описание реализации удалённого RMI-объекта на языке Java

В данном подпункте приводится описание удалённого объекта, реализующего интерфейс ***Rlp2Rmi***, который в последующем будет регистрироваться на общем сервере брокера запросов ***rmiregistry***. Исходный текст описания удалённого объекта представлен классом, представленном на листинге 3.2. Дополнительно учитываются следующие условия:

1) удалённый объект имеет имя ***Klp2RmiImpl***;
2) расширяет класс ***java.rmi.server.UnicastRemoteObject***;
3) сохраняет массив строковых объектов с именем ***msgs***, содержащий три элемента: ***количество*** полученных сообщений, ***последнее*** сообщение и ***предыдущее*** полученное сообщение.

*Листинг 3.2 – Исходный текст класса Klp2RmiImpl*

```
package klp2.rmi;

import java.rmi.RemoteException;

public class Klp2RmiImpl extends
      java.rmi.server.UnicastRemoteObject
      implements Klp2Rmi
{
      /**
       * Версия реализации
       */
      private static final long serialVersionUID = 1L;
      /**
       * Сохраняемые классом данные
       */
      private int N = 0;                      // Количество принятых сообщений
      private String[] msgs = {"0", "", ""}; // Сохраняемые сообщения

      // Конструктор
      protected Klp2RmiImpl()
            throws RemoteException
      {
            super();
      }
      /**
       * Реализация заявленного интерфейса
       */
      public String[] getMessage(String str)
            throws RemoteException
      {
            msgs[0] = Integer.toString(++N);
            msgs[2] = new String(msgs[1]);
            msgs[1] = new String(str);
            System.out.println(msgs[0] + ":" + msgs[1] + ":" + msgs[2]);

            return msgs;
      }
}
```

Данным исходным текстом завершается описание реализации удалённого RMI-объекта на языке Java.

**Замечание** — В следующем пункте данного пособия даётся ***описание реализации*** целевого сервера.

## 3.2.2 Реализация RMI-сервера

Вторая часть лабораторного задания №1 содержит описание Java-класса, который программно ассоциируется с реализацией RMI-сервера.

**Основная задача RMI-сервера** — регистрация на сервере ***rmiregistry*** всех удалённых объектов, обслуживаемых приложением-сервером.

В общем случае, RMI-сервер может регистрировать множество удалённых объектов, определённых разными интерфейсами.

В нашем случае, RMI-сервер будет регистрировать на сервере ***rmiregistry*** удалённый объект заданный классом ***Klp2RmiImpl*** и интерфейсом ***Klp2Rmi***.

Чтобы выполнить эту часть задания, студент должен:

1) ***подготовить*** инструментальную среду языка Java, включая каталоги размещения исходных текстов приложений сервера и клиента;
2) ***написать*** и правильно разместить исходные тексты всех частей серверной части приложения, включая последующую их компиляцию;
3) ***произвести*** последовательный запуск ***rmiregistry*** и RMI-сервера, убедившись в отсутствии ошибочных сообщений программного обеспечения.

Далее в следующих трёх пунктах показана реализация всех указанных действий, применительно к рассматриваемому в данном пособии примеру.

### 3.2.2.1 Подготовка среды реализации RMI-сервера

Подготовку среды реализации приложений сервера и клиента студент должен осуществлять при следующих ограничивающих требованиях, что демонстрируется для примера операционной среды ***ОС Linux*** и от имени пользователя ***vgr***:

1) сервер ***rmiregistry*** будет запускаться по умолчанию: по сетевому адресу ***localhost*** и порту ***1099***;
2) исходные тексты всех компонент приложения RMI-сервера размещаются в каталоге ***/home/vgr/rmiserver***;
3) исходные тексты всех компонент приложения RMI-клиента размещаются в каталоге ***/home/vgr/rmiclient***;
4) все действия студента по компиляции и запуску программного обеспечения будут осуществляться в виртуальных терминалах (в среде командной строки).

Исходя из указанных ограничений и теоретического материала, изложенного в разделе 2 учебного пособия «Распределённые вычислительные сети», настройка инструментальной среды языка Java сводится к заданию двух ключевых переменных среды ОС: ***JAVA_HOME*** и ***CLASSPATH***.

Рекомендуется разместить определение этих переменных в базовом файле конфигурации ***.bashrc***, например, в виде записей:

export JAVA_HOME=/usr/lib/jvm/default
export CLASSPATH=$HOME/rmiserver:$CLASSPATH
export CLASSPATH=$HOME/rmiclient:$CLASSPATH

**Замечание** — При использовании операционной среды ОС MS Windows, студенту следует соответствующим образом модифицировать экспорт указанных переменных.

#### 3.2.2.2 Компиляция исходного текста RMI-сервера

Исходный текст приложения RMI-сервера представлен на листинге 3.3. В представленном тексте использованы следующие дополнительные обозначения:

1) класс RMI-сервера представлен именем ***Klp2RmiServer***;
2) удалённый объект регистрируется под именем ***KLP2RMI***.

*Листинг 3.3 – Исходный текст класса Klp2RmiServer*

```
package klp2.rmi;

import java.rmi.Naming;

public class Klp2RmiServer
{

	public static void main(String[] args) {
		System.out.println("Klp2RmiServer: стартует... ");

		try {
			/**
			 * Создается экземпляр удаленного объекта класса Klp2RmiImpl.
			 */
			Klp2RmiImpl impl =
				new Klp2RmiImpl();

			/**
			 * Регистрация экземпляра удаленного объекта
			 * на сервере rmiregistry.
			 */
			Naming.rebind("//localhost:1099/RLP2RMI",(Klp2Rmi)impl);

			System.out.println("Klp2RmiServer: стартовал... ");
		}
		catch (Exception e)
		{
			System.out.println("Исключение: " + e);
		}

	}

}
```

Согласно заданным условиям, все компоненты RMI-сервера, определённые листингами 3.1, 3.2, 3.3 и включающие файлы ***Klp2Rmi.java***, ***Klp2RmiImpl.java***, ***Klp2RmiServer.java***, помещаются в каталог ***/home/vgr/rmiserver***, как это показано на рисунке 3.1.

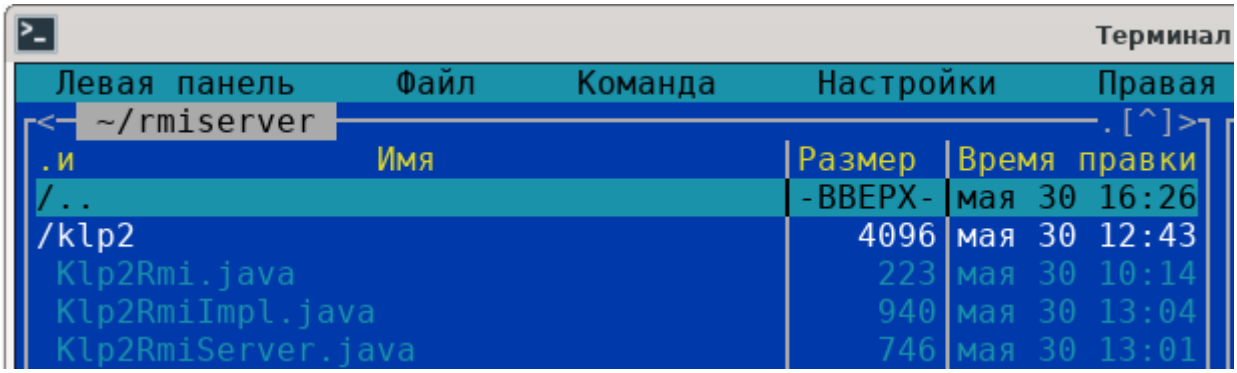


Рисунок 3.1 — Размещение исходных текстов компонент RMI-сервера

#### 3.2.2.3 Запуск и тестирование работы RMI-сервера

**Замечание** — Исходные тексты компонент RMI-сервера требуют правильной последовательности компиляции исходных текстов и правильного размещения полученных результатов.

Пошаговые действия правильной компиляции и размещения файлов, следующие.

**Шаг 1**. Создаются каталоги ***~/rmiserver/klp2*** и ***~/rmiserver/klp2/rmi***.

**Шаг 2**. Находясь в каталоге ***~/rmiserver***, проводится компиляция файла интерфейса командой ***javac Klp2Rmi.java***, после чего полученный после компиляции файл ***Klp2Rmi.class*** перемещается в каталог ***~/rmiserver/klp2/rmi***.

**Шаг 3**. Находясь в каталоге ***~/rmiserver***, проводится компиляция файла класса командой ***javac Klp2RmiImpl.java***, затем полученный файл ***Klp2RmiImpl.class*** перемещается в каталог ***~/rmiserver/klp2/rmi***.

**Шаг 4**. Находясь в каталоге ***~/rmiserver***, проводится компиляция файла класса командой ***javac Klp2RmiServer.java***, затем полученный файл ***Klp2RmiServer.class*** перемещается в каталог ***~/rmiserver/klp2/rmi***.

**Замечание** — В результате указанных пошаговых действий в каталоге ***~/rmiserver/klp2/rmi*** правильно размещаются все компоненты приложения RMI-сервера.

Рабочий запуск RMI-сервера осуществляется в два этапа.

**Этап 1**. Запускается виртуальный терминал, в котором выполняется команда ***rmiregistry***. В результате:

1) сервер ***rmiregistry*** запускается по умолчанию ***localhost:1099***;
2) сервер ничего не выводит;
3) терминал блокируется от ввода новых команд.

**Этап 2**. Запускается виртуальный терминал, в котором выполняется команда ***java klp2.mir.Klp2RmiServer***. В результате:

1) RMI-сервер проводит регистрацию удалённого объекта класса ***KlpRmiImpl*** по адресу ***localhost:1099/KLP2RMI***;
2) терминал блокируется от ввода новых команд;

3) при правильном запуске, сервер выводит в терминале информацию, показанную на рисунке 3.2.

```
Терминал
[vgr@upkasu ~]$ mc

[vgr@upkasu rmiserver]$ java klp2.rmi.Klp2RmiServer
Klp2RmiServer: стартует...
Klp2RmiServer: стартовал...
```

Рисунок 3.2 — Нормальный запуск RMI-сервера в виртуальном терминале

На этом этапе, все работы по лабораторному заданию №1 считаются успешно выполненными.

Студенту следует отразить полученные результаты собственного исследования в едином личном отчёте.

**Замечание** — Выполненное выше задание предполагает использование ОС УПК АСУ, которое установлено в учебных классах кафедры АСУ. В следующем пункте приводится учебный материал, позволяющий выполнить это же задание в среде ОС MS Windows 10.

### 3.2.3 Запуск RMI-сервер в среде ОС MS Windows 10

Как было отмечено ранее, при старте RMI-сервер регистрирует свой интерфейс на сервере ***rmiregistry***. При этом, rmiregistry должен быть доступен интерфейс приложения. В нашем случае — это файл ***Klp2Rmi.class***.

Чтобы всё работало правильно, нужно правильно настроить переменную среды ***CLASSPATH***. Обычно неправильная настройка этой переменной является источником ошибок начинающих программистов на языке Java.

Данный подраздел ориентирован на пользователей ОС MS Windows 10 и изложен в трех пунктах:

а) инсталляция JDK версии 17.02;

б) настройка переменных среды;

в) компиляция и старт RMI-сервера.

#### 3.2.3.1 Инсталляция JDK версии 17.02

С сайта www.oracle.com можно скачать «Установщик x64», как это показано на рисунке 3.3.

www.oracle.com Загрузки Java | Oracle ★30 отзывов

ORACLE Продукты ОТРАСЛИ Ресурсы Поддержка Мероприятия Разработчик Партнеры

JDK 17 будет получать обновления в соответствии с этими условиями, по крайней мере, до сентября 2024 года.

**Загрузок Java SE Development Kit 17.0.2**

Благодарим вас за загрузку этого выпуска платформы Java™, комплекта разработки Standard Edition (JDK™). JDK-это среда разработки для создания приложений и компонентов с использованием языка программирования Java.

JDK включает в себя инструменты для разработки и тестирования программ, написанных на языке программирования Java и работающих на платформе Java.

**Линукс** macOS **Windows**

| Описание продукта/файла | Размер файла | Скачать |
|---|---|---|
| Сжатый архив x64 | 171,34 МБ | https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256) |
| Установщик x64 | 152,43 МБ | https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256) |
| Установщик x64 MSI | 151,32 МБ | https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256) |

Рисунок 3.3 — Сайт с дистрибутивом JDK версии 17.02

Запустив установщик, следует выбрать каталог дистрибутива или установить JDK по умолчанию.

Далее, исключительно для учебных целей, дистрибутив установлен в каталог ***C:\Java***.

После установки дистрибутива, переходим в каталог ***C:\Java*** и создаём там два каталога, как показано на рисунке 3.4:

а) ***rmiserver*** для запуска сервера;

б) ***rmiclient*** для запуска программы клиента.

c [windows10] 59 330 952 Кб из 101 820 412 Кб свободно

c:\Java*.*

| Имя | Тип | Размер | Дата | Атриб |
|---|---|---|---|---|
| [..] | | <Папка> | 27.01.2022 16:23 | — |
| [bin] | | <Папка> | 27.01.2022 16:20 | — |
| [conf] | | <Папка> | 27.01.2022 16:20 | — |
| [include] | | <Папка> | 27.01.2022 16:20 | — |
| [jmods] | | <Папка> | 27.01.2022 16:21 | — |
| [legal] | | <Папка> | 27.01.2022 16:21 | — |
| [lib] | | <Папка> | 27.01.2022 16:21 | — |
| [rmiclient] | | <Папка> | 27.01.2022 22:38 | — |
| [rmiserver] | | <Папка> | 27.01.2022 16:50 | — |
| COPYRIGHT | | 3 244 | 07.12.2021 22:06 | -a- |
| LICENSE | | 6 186 | 27.01.2022 16:20 | -a- |
| README | | 290 | 27.01.2022 16:20 | -a- |
| release | | 1 263 | 27.01.2022 16:21 | -a- |

Рисунок 3.4 — Содержимое каталога C:\Java\

**Замечание** — Обычно разработку программного обеспечения не ведут в каталогах дистрибутивов, но для учебных целей это — приемлемо.

#### 3.2.3.2 Настройка переменных среды

Для настройки переменных среды следует воспользоваться соответствующими инструментальными средствами ОС:

а) на нижней панели рабочего стола следует активировать кнопку главного меню правой кнопкой мыши и, в появившемся контекстном меню, выбрать пункт «**Система**»;
б) в появившемся окне «Система» следует активировать пункт меню «**Дополнительные параметры системы**»;
в) в появившемся окне «Свойства системы» необходимо активировать кнопку «**Переменные среды...**»; в результате появится окно «**Переменные среды**», которое показано на рисунке 3.5.

Редактированию подвергаются две переменные CLASSPATH и Path, что отмечено стрелками на рисунке 3.5:

а) если переменная CLASSPATH – отсутствует, то её нужно создать с помощью кнопки «**Создать...**»;
б) в переменную Path необходимо добавить каталог **C:\Java\bin**; тогда все утилиты языка Java будут запускаться без указания абсолютного пути; в частности, это касается сервера ***rmiregistry***;
в) после завершения установки переменных среды, все окна терминалов необходимо закрыть и запустить снова.

#### 3.2.3.3 Компиляция и старт RMI-сервера

Перед компиляцией и запуском RMI-сервера следует открыть окно терминала (Командная строка) и запустить ***rmiregistry***.

После проведённой настройки переменных среды, достаточно указать только имя команды. В частности, если ОС обнаружит проблемы с безопасностью, то при запуске ***rmiregistry*** могут выдаваться различные предупреждающие сообщения.

Далее считаем, что исходные тексты RMI-сервера размещены в каталоге ***C:\Java\rmiserver***, как это показано на рисунке 3.6.

Обратите внимание, что в каталог с исходными текстами RMI-сервера добавлен файл сценария **RunS.cmd**, содержимое которого показано на рисунке 3.7.

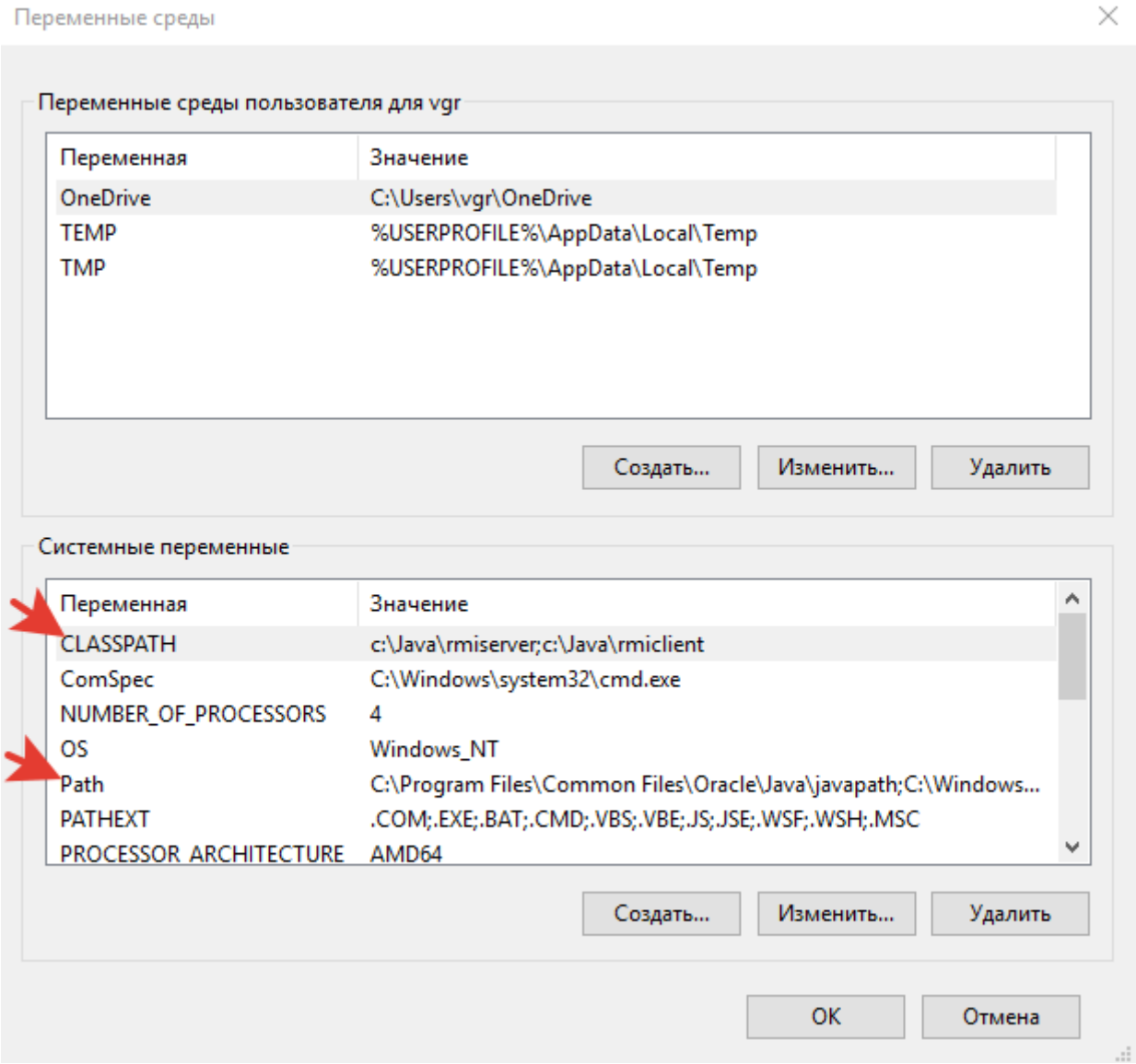


Рисунок 3.5 — Окно редактирования переменных среды Windows 10

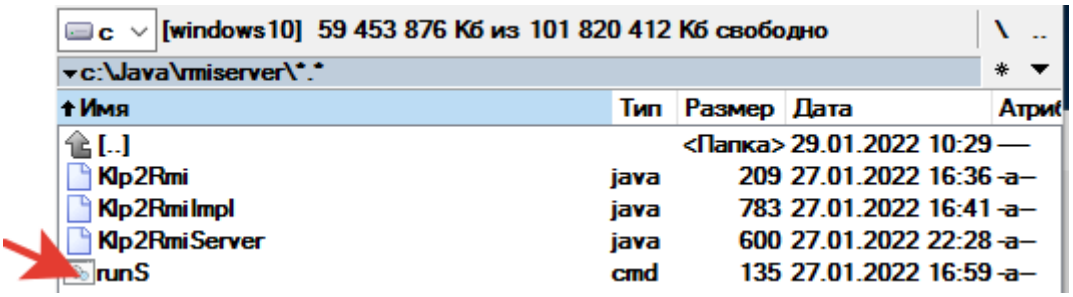


Рисунок 3.6 — Каталог для запуска RMI-сервера

runS — Блокнот

Файл Правка Формат Вид Справка

```
javac -d ./ Klp2Rmi.java
javac -d ./ Klp2RmiImpl.java
javac -d ./ Klp2RmiServer.java

java -classpath ./ klp2.rmi.Klp2RmiServer
```

Рисунок 3.7 — Содержимое файла сценария RunS.cmd

Обратите внимание, что три команды **javac** последовательно компилируют исходные тексты RMI-сервера и размещают их в нужных каталогах. Это достигается с помощью опции «**-d**», после чего указывается каталог, относительно которого будет рассматриваться оператор ***package*** в исходных текстах программ RMI-сервера.

Последняя команда **java** производит запуск RMI-сервера, причём используемая опция также указывает каталог, относительно которого будут искаться классы запускаемой программы.

Для нашего случая опцию «**-classpath**» можно не использовать, поскольку сам запуск проводится в пределах проведённых ранее настроек.

**Замечание** — Если неправильно настроить переменные среды, как это указано в предыдущем пункте, то RMI-сервер не запустится, поскольку **rmiregistry** не найдёт класс **Rlp2Rmi.class** и откажет серверу в регистрации.

## 3.3 Пример выполнения лабораторного задания №2

**Лабораторная работа №2** содержит описание и реализацию примера ***клиентской части распределенного приложения***, демонстрирующего применение технологии RMI, входящей в состав пакета Java SE: JDK SE (инструментального пакета языка Java уровня Standard Edition).

> **Методически**, лабораторное задание №2 является продолжением разработки распределенного приложения, серверная часть которого уже должна быть реализована в лабораторной работе №1.

Структура и последовательность работ, описанных в данном подразделе, следует структуре предыдущего подраздела и также разделена на две части:

1) описание функциональности примера приложения-клиента;
2) пример реализации RMI-клиента.

### 3.3.1 Формулировка задания на выполнение работы №2

В общем случае приложение-клиент может быть слабо связано с функциональностью приложения-сервер. Это подразумевает, что приложение клиента может:

1) *использовать* множество удалённых объектов, размещённых на множестве серверов приложений, которые функционально и организационно никак не связаны между собой;
2) *включать* очень маленькую функциональность разных отдельных удалённых объектов по сравнению с собственной функциональностью;
3) *учитывать* только синтаксическое описание удалённых объектов, которое предоставлено в виде их интерфейсов.

> **Замечание** — В рамках выполнения лабораторной работы, студенту не следует сильно расширять функциональность своего приложения-клиента. Достаточно продемонстрировать адекватное использование методов, уже описанных во время выполнения задания №1.

#### 3.3.1.1 Функциональное описание конкретного RMI-клиента

Для учебного примера функциональность приложения-клиента задано в виде двух условий:

1) клиент ***посылает*** серверу текстовую строку, используя метод ***getMessage(...)*** удалённого объекта;
2) клиент выводит на печать результат обращения к методу ***getMessage(...)***.

#### 3.3.1.2 Описание интерфейса RMI-клиента

Считается, что описание удалённого интерфейса, содержащего синтаксис метода ***getMessage(...)***, соответствует исходному тексту языка Java, представленному ранее на листинге 3.1, подпункта 3.2.1.2, подраздела 3.2 данного пособия.

> **Замечание** — Поскольку разработчику приложения-клиента обычно доступен только исходный текст интерфейса удалённого объекта, то допустимо модифицировать оператор ***package*** для согласования размещения программного обеспечения клиента.

### 3.3.2 Реализация RMI-клиента

Вторая часть лабораторного задания №2 содержит описание Java-класса, который программно ассоциируется с реализацией приложения RMI-клиента.

> **Основная задача RMI-клиента** — подключение ко всем нужным серверам ***rmiregistry*** с целью поиска всех нужных точек подключения к удаленным объектам и выполнения обращений к этим объектам.

В нашем случае, на уже запущенном сервере ***rmiregistry***, должен быть зарегистрирован удалённый объект заданный классом ***Klp2RmiImpl*** и интерфейсом ***Klp2Rmi***. Дополнительно, должна быть подготовлена среда реализации RMI-клиента, что уже было выполнено в предыдущем задании и подробно описано в подпункте 3.2.2.1 данного пособия.

**Более конкретно**, выполняя эту часть своего задания, студент ориентируется на реализацию своего серверного приложения и организацию его среды разработки и запуска. Далее, - необходимо:

1) ***написать*** и правильно разместить исходные тексты всех частей ***клиентской части*** приложения, включая последующую их компиляцию;
2) ***произвести*** необходимое количество запусков приложения RMI-клиента, убедившись в отсутствии ошибочных сообщений программного обеспечения, и демонстрируя работоспособность всего распределенного приложения.

> **Замечание** — Далее, на заданном учебном примере, демонстрируются все необходимые действия.

#### 3.3.2.1 Компиляция исходного текста RMI-клиента

**Исходный текст** учебного примера приложения RMI-клиента, удовлетворяющего перечисленным выше ограничениям, представлен на листинге 3.4.

**Дополнительные ограничения** на реализацию этого текста представлены следующими условиями:

1) RMI-клиент реализован как класс ***Klp2RmiClient***;
2) удалённый объект клиента доступен по ссылке: ***//localhost:1099/KLP2RMI***;
3) передаваемая на сервер строка задаётся аргументом приложения.

*Листинг 3.4 – Исходный текст класса Klp2RmiClient*

```
package klp2.rmi;

import java.rmi.Naming;

public class Klp2RmiClient
{

      public static void main(String[] args)
      {
            /**
             * Читаем и устанавливаем значение аргумента
             */
            String str = "";  // Строка аргумента
            if(args.length > 0) str = new String(args[0]);
            System.out.println("Клиент: число аргументов " + args.length);
            /**
             * Результат обращения к серверу
             */
            String[] msgs;

            try{
              /**
               * Получаю объектную ссылку на удаленный объект
               */
              Klp2Rmi rmiobj =
                        (Klp2Rmi)Naming.lookup("//localhost:1099/KLP2RMI");

              System.out.println("Посылаю: " + str);
              /**
                * Вызываю удалённый метод
                */
              msgs =
                        rmiobj.getMessage(str);

              /**
                * Печатаю результат работы приложения-клиента
                */
              System.out.println("Клиент: число сообщений сервера: " + msgs[0]);
              System.out.println("Последнее     сообщение сервера: " + msgs[1]);
              System.out.println("Предпоследнее сообщение сервера: " + msgs[2] +
                  "\n");
          }
          catch (Exception e)
          {
                System.out.println("Client-ERROR : " + e) ;
          }
      }
}
```

**Согласно условиям компиляции и запуска** компонент распределенного приложения, исходный текст листинга 3.4 должен быть помещён в каталог указанный в системной переменной ***CLASSPATH***, в котором также должны быть подготовлены подкаталоги, соответствующие оператору ***package*** исходного теста программы (см. рисунок 3.8).

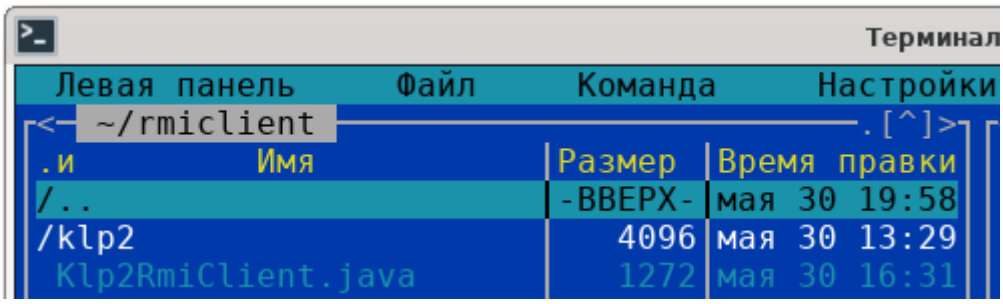


Рисунок 3.8 - Размещение исходного текста приложения RMI-клиента

Компиляция исходного текста RMI-приложения осуществляется в этом же каталоге, командой: ***javac Klp2RmiClient.java***

Полученный результат, в виде файла ***Klp2RmiClient.class***, помещается в каталог: ***~/rmiclient/klp2/rmi***.

**Замечание** — В случае, когда RMI-сервер запускается на другой ЭВМ, необходимо дополнительно в каталог ***~/rmiclient*** поместить исходный текст файла-интерфейса приложения: ***Klp2Rmi.java***. Тогда, сначала необходимо провести компиляцию и размещение файла-интерфейса, как это описано в пункте 1.2.3 настоящего пособия, а затем — переходить к компиляции приложения-клиента.

### 3.3.2.2 Запуск и тестирование работы RMI-клиента

Запуск приложения-клиента осуществляется из каталога ***~rmiclient*** командой общего вида: ***java klp2.rmi.Klp2RmiClient Сообщение***

Пример такого запуска приведён на рисунке 3.9.

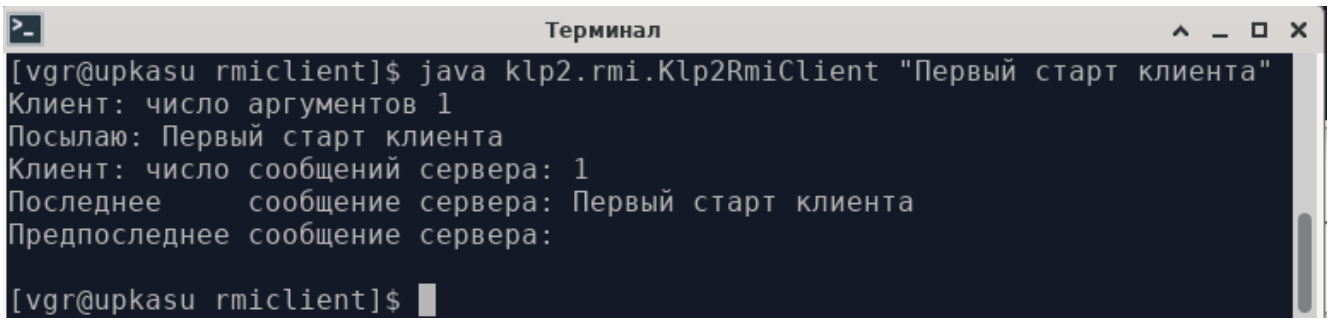


Рисунок 3.9 — Пример запуска приложения RMI-клиента

### 3.3.3 Запуск RMI-клиента в среде ОС MS Windows 10

Поскольку все необходимые переменные среды уже были настроены в предыдущем разделе, то здесь мы ограничимся следующими подсказками:

а) исходные тексты программы клиента размещены в каталоге **C:\Java\rmiclient**, как это показано на рисунке 2.3;

б) сценарий запуска RMI-клиента показан на рисунке 3.10.

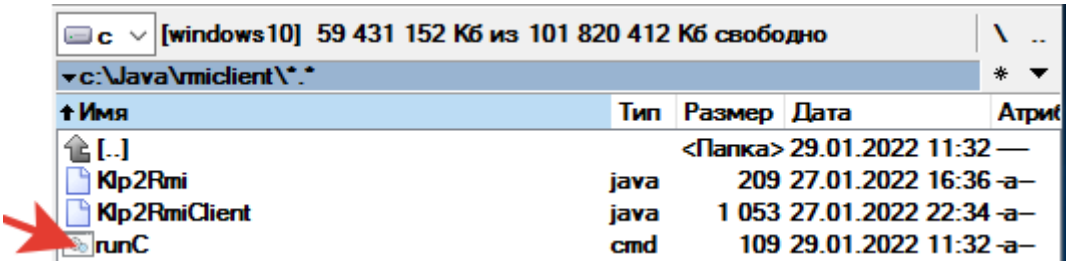


Рисунок 2.3 — Каталог для запуска RMI-клиента

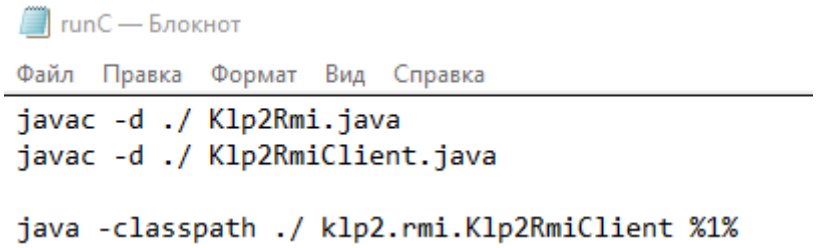


Рисунок 3.10 — Содержимое файла сценария RunC.cmd

**Замечание** — Поскольку программа RMI-клиента запускается много раз, то, после первого её запуска, все команды ***javac*** можно закомментировать оператором ***REM***.

## 3.4 Оценка выполнения студентом сессии ЛЭС

**Оценка** выполненного студентом индивидуального задания лабораторно-экзаменационной сессии (ЛЭС) осуществляется преподавателем кафедры АСУ ТУСУР, который заявлен и согласован с руководством ФДО ТУСУР.

**Общее задание** на выполнение работ ЛЭС формулируется следующим образом:

1. Каждый студент должен реализовать своё собственное распределённое приложение, по аналогии примеров, приведённых в подразделах 3.2 и 3.3 данного пособия.
2. Результат реализации своего проекта, а также его отладку и тестирование, студент должен письменно изложить в своём едином личном отчёте.
3. Итоговый вариант отчёта должен быть размещён для оценивания в электронном журнале дисциплины (на сетевом диске СДО ФДО), в месте и сроках объявленных преподавателем во время проведения ЛЭС.

**По результатам** отчёта выполненных лабораторных работ №1 и №2 студент проходит индивидуальное собеседование с преподавателем и оценивается в виде дифференцированного зачёта (***зачёта с оценкой***).

**Результат оценивания** студента заносится преподавателем в электронный журнал дисциплины и сообщается руководству ФДО ТУСУР, в обычном для промежуточной аттестации порядке.